Randomized algorithms for determining the majority on graphs *

Gianluca De Marco † Andrzej Pelc ‡

Abstract

Every node of an undirected connected graph is colored white or black. Adjacent nodes can be compared and the outcome of each comparison is either 0 (same color) or 1 (different colors). The aim is to discover a node of the majority color, or to conclude that there is the same number of black and white nodes. We consider randomized algorithms for this task and establish upper and lower bounds on their expected running time. Our main contribution are lower bounds showing that some simple and natural algorithms for this problem cannot be improved in general.

keywords: graph, majority problem, randomized algorithm.

*A preliminary version of this paper appeared in the Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'2003), August 2003, Bratislava, Slovakia, LNCS 2747, 368-377.

†Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, via Moruzzi 1, 56124 Pisa, Italy (E-mail: gianluca.demarco@iit.cnr.it). This work was done during this author's stay at the Research Chair in Distributed Computing of the Université du Québec en Outaouais.

‡Département d'informatique, Université du Québec en Outaouais, Gatineau, Québec J8X 3X7, Canada (E-mail: andrzej.pelc@uqo.ca). The research of this author was partially supported by NSERC grant OGP 0008136 and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais.

1 Introduction

Given an undirected connected n-node graph $G = (V, E)$, any assignment of colors *white* or *black* to the nodes of G such that there are at most $n/2$ black nodes, is referred to as a *coloring* of G. Given such a coloring, adjacent nodes can be compared and the outcome of each comparison is either 0 (same color) or 1 (different colors). The aim is to discover a white node, in the case when white nodes form a strict majority, or else to conclude that there is the same number of black and white nodes, using as few comparisons as possible.

This problem has been investigated in [2, 3, 10] for the complete graph, i.e., in the case when any pair of nodes can be compared. It has been proved in [10, 2] that the minimum worst-case number of comparisons to deterministically solve this problem is $n - \nu(n)$, where $\nu(n)$ is the number of 1-bits in the binary representation of n. In [3] the minimum average-case number of comparisons was investigated for this problem.

The above problem has a similar flavor to that of diagnosis of multiprocessor systems, introduced in [9]. Such a system is represented by an undirected connected graph whose nodes are processors. Some processors are faulty, there are less than one-half of such processors, and the aim of diagnosis is to locate all faulty processors, by performing tests on some processors by adjacent ones. Among many fault models considered in the literature ([9] introduced the first of them, called the PMC model), two are particularly relevant in our context: the symmetric comparison model of Chwa and Hakimi [4], and the asymmetric comparison model of Malek [8]. In both models, comparison tests can be conducted between adjacent processors. A comparison test between two fault-free processors gets answer 0 (no difference) and the comparison test between a fault-free and a faulty processor gets answer 1 (difference). This is also identical to our assumptions (black nodes representing faulty processors). The two models differ between them and also differ from our setting when two faulty processors are compared. In the symmetric comparison model, the answer can be then arbitrary (0 or 1) and in the asymmetric comparison model the answer is 1. The justification usually given for the two above diagnosis models is the following. Comparison tests often consist in choosing a fairly complex computational task and submitting it to both compared processors. If the results of computations are identical for both processors, the answer to the test is 0, if not, the answer is 1. Two fault-free processors will clearly give the same result of the computation, and a faulty processor will likely make an error somewhere in the computation, thus producing a different result than a good one. The situation is less clear when two faulty processors are compared. They may either err in the same way, thus causing answer 0 to the comparison test, or make different mistakes, the test answer being 1 in this case. This argument justifies the symmetric model. However, one may say that, for a complex computational task, identical errors for two faulty processors are very unlikely, thus the asymmetric comparison model could be more realistic.

Our testing model, in which comparison test results faithfully describe the same or different fault status of tested processors (same or different node colors) can be justified by another

scenario. Suppose that all processors of the system have a boolean variable identically initialized in the beginning. Then some processors (less than half of them) fail, and the fault consists in corrupting precisely this bit. We want to discover the original bit (which corresponds to the majority color, since only less than half of the processors changed it) by comparing the value of this boolean variable between adjacent processors. This situation is similar to the persistent bit problem described in [7], although the focus in [7] was to distributedly restore the common bit in all nodes, using only local probes of the network by each processor.

In the above context of fault-tolerance, it is natural to assume that faulty processors (black nodes) are significantly less numerous than the good ones (white nodes), since in realistic systems the number of faults rarely approaches 50%. Therefore, it is reasonable to suppose that the number of black nodes is at most αn, for some positive constant $\alpha < 1/2$. A coloring satisfying this assumption will be called an *α-coloring*. Thus an α-coloring of an n-node graph $G = (V, E)$ is a function $I : V \to \{b, w\}$, (where b and w stand for black and white, respectively), with $|I^{-1}(\{b\})| \leq \alpha n$.

We now formulate two variations of the problem considered in this paper.

The simple-majority problem on graphs (MPG).

> Let $G = (V, E)$ be an undirected connected graph with an input coloring defined on it. If white nodes strictly outnumber black nodes, we must discover a white node $v \in V$, and report equality otherwise, by making comparisons beetwen adjacent nodes of G. The outcome of every comparison is either 0 (equal colors) or 1 (nonequal colors). The goal is to use as few comparisons as possible.

The α-majority problem on graphs (α-MPG).

> Let $G = (V, E)$ be an undirected connected graph with an input α-coloring defined on it, for some $\alpha < 1/2$. We must discover a white node $v \in V$ by making comparisons beetwen adjacent nodes of G. The outcome of every comparison is either 0 (equal colors) or 1 (nonequal colors). The goal is to use as few comparisons as possible.

For both these problems, we refer to the number of comparisons used by an algorithm on a given input coloring (resp. α-coloring), as the running time of the algorithm on this input. Hence we assume that all operations other than tests take negligible time.

It should be noted that in the α-MPG, the parameter $\alpha < 1/2$ is known to the algorithm selecting comparisons. Aigner [1] considered this variation of the majority problem in the deterministic setting for the complete graph, and proved that the minimum number of comparisons is $2\lfloor \alpha n \rfloor - \nu(\lfloor \alpha n \rfloor)$. Obviously, for any connected graph, $2\lfloor \alpha n \rfloor$ comparisons are

sufficient, by performing tests along edges of a spanning tree of any connected subgraph with $2\lfloor \alpha n \rfloor + 1$ nodes. Aigner [1] also pointed out interesting relations between the α-MPG and the diagnosis problem in the PMC model from [9], in the case of the complete graph.

Hence (asymptotically) optimal solutions to both MPG and α-MPG are known in the deterministic setting. Since the running time of optimal algorithms is linear in the number of nodes in both cases, it is natural to seek randomized algorithms for both problems, hoping to improve efficiency. Thus in the present paper we concentrate on randomized algorithms for both these problems. It turns out that while randomization does not help significantly in the case of the MPG, it sometimes drastically improves algorithm complexity for the α-MPG. Our main contribution are lower bounds showing that the complexity of some simple and natural algorithms for these problems cannot be improved in general.

1.1 Our results

We first show that the simple-majority problem does not allow efficient randomized algorithms on any graphs. Indeed, we prove that if the difference between the number of white and black nodes is bounded by a constant then every randomized algorithm for determining a white node in an n-node graph (with sufficiently small constant error probability) uses expected running time $\Omega(n)$ on some input, even for the complete graph. (As mentioned above, $O(n)$-time algorithms – even deterministic – exist for any connected graph). Hence, in the rest of the paper we investigate randomized algorithms for the α-majority problem on n-node connected graphs, with parameter $\alpha < 1/2$. We study expected running time of randomized Monte Carlo algorithms for the α-MPG, whose probability of error is at most $\epsilon > 0$. For any connected graph, we show an algorithm whose running time on every input is $O(D \log(1/\epsilon))$, where D is the diameter of the graph. If the maximum degree of a graph is βn, where $\beta > 2\alpha$, then there is an algorithm with running time $O(\log(1/\epsilon))$. We show that these bounds cannot be improved in general. Every algorithm, running on an arbitrary n-node graph, must use expected time $\Omega(\min(n, \log(1/\epsilon)))$ on some input. We also show that the large constant β in the requirement of maximum degree of βn is essential to get a fast algorithm: we show graphs of maximum degree $\Theta(n)$, for which every algorithm must use expected time $\Omega(n)$ on some input. On the other hand, for sufficiently small constant ϵ, the bound $O(D)$ cannot be improved for a large class of graphs: we show that, for d-dimensional grids and tori, every algorithm must use expected time $\Omega(D)$ on some input.

2 Terminology and preliminaries

Throughout the paper, we restrict attention to connected graphs. Given a graph $G = (V, E)$ and an input coloring I on it, we define the *comparison function $\mathcal{L}_I$ of G on input I* as the function $\mathcal{L}_I : E \to \{0, 1\}$, such that for each edge $e = \{u, v\} \in E$,

$$\mathcal{L}_I(e) = \begin{cases} 0 & \text{if } \ I(u) = I(v); \\ 1 & \text{otherwise.} \end{cases}$$

We will use deterministic algorithms as a tool to prove lower bounds on the performance of randomized algorithms. Fix a *deterministic* algorithm A for the MPG (resp. α-MPG). Given a graph $G = (V, E)$ and an input coloring (resp. α-coloring) I on it, algorithm A works in consecutive steps defined as follows. At any step, the algorithm selects an edge $e \in E$ and receives as an answer $\mathcal{L}_I(e)$. After the last step, it shows a node of the graph as the solution (a discovered white node).

For any input I, the set $E_I \subseteq E$ denotes the set containing all the edges selected during the execution of A. The *running time* of algorithm A on input I is $r(I) = |E_I|$. For any input I and integer γ, we define the set $E_I(\gamma)$ as follows

$$E_I(\gamma) = \begin{cases} \{_{e \in E_I|\ e \text{ is selected at step } s \leq \gamma}\} & \text{if } \gamma < |E_I| \\ E_I & \text{otherwise.} \end{cases}$$

Given an input I, we define the *execution* $\mathcal{E}(A, I)$ of algorithm A on input I, as the pair $\mathcal{E}(A, I) = (G_I, u_I)$, where $G_I = (V_I, E_I)$ is the subgraph of G induced by the set of edges E_I, and u_I is the node representing the solution given by A.

We will use the following version of Chernoff bound.

Lemma 2.1 (Chernoff bound [6]) *Let X be the number of successes in a series of Bernoulli trials of length m with success probability q. Let $q' < q$. Then $Prob(X \leq q'm) \leq e^{-am}$, where a is a positive constant depending on q and q' but not on m.*

The main tool for proving lower bounds on the performance of randomized algorithms will be the following well known result.

Lemma 2.2 (Yao's minimax principle [11]) *Let $0 < \epsilon < 1/2$. For any probability distribution $\mathcal{P}$ over the set of inputs, let $\mathcal{A}(\mathcal{P})$ denote the set of all deterministic algorithms that err with probability at most 2ϵ over $\mathcal{P}$. For $A \in \mathcal{A}(\mathcal{P})$, let $C(A, \mathcal{P})$ denote the expected running time of A over $\mathcal{P}$. Let $\mathcal{R}$ be the set of randomized algorithms that err with probability at most ϵ for any input, and let $T(R, I)$ denote the expected running time of $R \in \mathcal{R}$ on input I. Then, for all $\mathcal{P}$ and all $R \in \mathcal{R}$,*

$$\min_{A \in \mathcal{A}(\mathcal{P})} C(A, \mathcal{P}) \leq 2 \max_I T(R, I).$$

The standard application of the above lemma to lower bound proofs is the following. We construct a probability distribution over the set of inputs for a given graph, for which any

deterministic algorithm that errs with probability at most 2ϵ has a large expected running time over this probability distribution. (Note that there is a big flexibility in the choice of the distribution, in fact any set of inputs can be selected, by putting probability zero on other inputs). In view of the lemma, this implies that every randomized (Monte Carlo) algorithm that errs with probability at most ϵ for any input, must have large expected running time on some input.

3 The simple-majority problem on graphs

In this section we show that the simple-majority problem on graphs does not allow efficient randomized algorithms. Indeed, we prove that if the difference between the number of white and black nodes is bounded by a constant then every randomized algorithm for determining a white node in an n-node graph (with sufficiently small constant error probability) uses expected running time $\Omega(n)$ on some input, even for the complete graph. Note that this lower bound holds even if the exact (constant) difference between the number of black and white nodes is known to the algorithm selecting comparisons. This lower bound on complexity is tight in view of the obvious algorithm using $n-1$ comparisons on any connected graph: performing all tests along edges of a spanning tree, and determining the majority color.

Theorem 3.1 *Consider an arbitrary graph $G=(V,E)$ and an arbitrary positive integer constant d. Suppose that the number of white nodes exceeds that of black nodes by d. Any randomized algorithm for determining a white node in G, which errs with probability at most ϵ, for sufficiently small constant ϵ, must have expected running time $\Omega(n)$ on some input I.*

Proof: Fix a positive integer constant d. Consider the set $\mathcal{I}$ of input colorings on G, for which the number of white nodes exceeds that of black nodes by d. We prove that any *deterministic* algorithm for the MPG on G, that errs with probability at most 2ϵ on the uniform distribution on $\mathcal{I}$, uses an expected number of $\Omega(n)$ comparisons. By Yao's minimax principle, this implies our theorem.

Let g be the size of the set $\mathcal{I}$. Let $c=10^d\cdot d!$ and fix $\epsilon<1/(4c+4)$. Fix a deterministic algorithm A for the MPG on G which errs with probability at most 2ϵ on the uniform distribution on $\mathcal{I}$. We will prove that $r(I)>n/10$ for at least $g/2$ inputs I.

Suppose not. Hence $r(I)\leq n/10$ for at least $g/2$ inputs I. Let $\mathcal{J}$ be the set of inputs I for which $r(I)\leq n/10$. Denote by $\mathcal{J}^+$ the set of inputs in $\mathcal{J}$ on which algorithm A is correct, and by $\mathcal{J}^-$ the set of inputs in $\mathcal{J}$ on which algorithm A is incorrect. Let V_I be the set of nodes involved in comparisons made by A on input I and let u_I be the node presented by A as a solution on input I. Denote $U_I=V_I\cup\{u\}$. We have $|U_I|\leq 2r(I)+1\leq 3n/10$. Denote by W_I the set of nodes outside of U_I which are colored white on input I. Since white nodes strictly outnumber black nodes, we have $|W_I|\geq n/5$, for any $I\in\mathcal{J}$. For any

$I \in \mathcal{J}^+$ and any set $Z \subseteq W_I$ of size d, let $f(I, Z)$ denote the coloring resulting from I by interchanging colors white and black at each node outside of Z. Notice that $f(I, Z) \in \mathcal{I}$ because the number of white nodes in $f(I, Z)$ exceeds that of black nodes by exactly d (the set Z is a set of those extra d white nodes).

For any $I \in \mathcal{J}^+$ and any $Z \subseteq W_I$ of size d, we have $\mathcal{E}(A, I) = \mathcal{E}(A, f(I, Z))$. In particular, $f(I, Z) \in \mathcal{J}$. Moreover, for any $I \in \mathcal{J}^+$, algorithm A is incorrect on $f(I, Z)$, and hence $f(I, Z) \in \mathcal{J}^-$. For a given $I \in \mathcal{J}^+$ and different sets Z_1, Z_2, the colorings $f(I, Z_1)$ and $f(I, Z_2)$ are different. On the other hand, for a fixed set Z and fixed coloring J, there is only one coloring I such that $J = f(I, Z)$. Consider the bipartite graph on the set $\mathcal{J}$ of colorings with bipartition $\mathcal{J}^+$, $\mathcal{J}^-$ and edges between those $I \in \mathcal{J}^+$ and $J \in \mathcal{J}^-$, for which $J = f(I, Z)$, for some set $Z \subseteq W_I$ of size d. Every $I \in \mathcal{J}^+$ has degree at least

$$\binom{n/5}{d} \geq \frac{(n/10)^d}{d!} = \frac{n^d}{c}.$$

Every $J \in \mathcal{J}^-$ has degree at most $\binom{n}{d} \leq n^d$. Hence

$$|\mathcal{J}^-| \geq \frac{n^d/c \cdot |\mathcal{J}^+|}{n^d} = |\mathcal{J}^+|/c.$$

This implies $|\mathcal{J}^+| \leq c|\mathcal{J}^-|$, hence $g/2 \leq |\mathcal{J}| \leq (c+1)|\mathcal{J}^-|$, and consequently $|\mathcal{J}^-| \geq g/(2c+2)$. Thus the probability that the algorithm is incorrect is at least $1/(2c+2) > 2\epsilon$, which is a contradiction.

Hence $r(I) > n/10$ for at least $g/2$ inputs I. This implies that the expected running time of algorithm A is $\Omega(n)$, and the theorem follows. □

4 Upper bounds for the α-majority problem on graphs

In this section we establish upper bounds on the expected running time of randomized algorithms for the α-MPG, and show efficient randomized algorithms for this problem on a large class of graphs. We begin with the following lemma.

Lemma 4.1 *Fix $\alpha < 1/2$. Let $\beta > 2\alpha$ and let H be a connected subgraph of an n-node graph G, with $k \geq \beta n$ nodes. Let $0 < \epsilon < 1$ be the bound on error probability. Then there exists a randomized algorithm for the α-MPG on the graph G which errs with probability at most ϵ and has running time $O(D \log(1/\epsilon))$ on every input, where D is the diameter of the graph H.*

Proof: Consider a series of $m = \lceil c \log(1/\epsilon) \rceil$ Bernoulli trials with success probability $q = (\beta - \alpha)/\beta$. Let F be the event that in this series there are at most $m/2$ successes. By Lemma

2.1, the probability of event F is at most ϵ, for some constant c, in view of $q > 1/2$. Fix such a constant c.

The algorithm works as follows.

1. Make m random independent selections of nodes in H with uniform probability $1/k$. Note that selections are with return, so it is possible to select the same node many times.
2. Let $S = \{s_1, ..., s_h\}$ be the set of selected nodes. Construct a spanning subtree T of this set in the graph H in the following greedy manner: let T_{i-1} be the part of T constructed till step $i-1$. (T_0 consists of node s_1). At step i, join node s_{i+1} to the closest (in H) among nodes in T_{i-1} by a shortest path in H.
3. Perform all tests along edges of this tree. Answers to these tests induce an assignment of colors 1 and 2 to all nodes of the tree. Consider colors assigned to nodes of S and count them with multiplicities, i.e., add x to the count of a given color if a node of this color was chosen x times in the selection.
4. Let $i \in \{1, 2\}$ be the color that got count at least as large as the other color (in the case of equality let $i = 1$). Select a node $v \in S$ of color i. Choose this node v as the solution (white) node.

In order to analyze this algorithm, observe that when its solution is incorrect then the majority among the m random selections must be black. This means that in the corresponding Bernoulli series with success probability q (success means selecting a white node) there are at most $m/2$ successes. By the choice of the constant c, this probability is at most ϵ, hence the algorithm errs with probability at most ϵ, as required. The total number of tests is at most $rD \leq mD \in O(D \log(1/\epsilon))$, for any input. □

Notice that the bound on running time holds for *every* execution of the algorithm (not only for the expected value of the running time), for every input.

Putting $H = G$ in the above lemma we obtain the following result.

Theorem 4.1 *Fix $\alpha < 1/2$. For any $0 < \epsilon < 1$ and any connected graph G there exists a randomized algorithm for the α-MPG, which errs with probability at most ϵ and has running time $O(D \log(1/\epsilon))$ on every input, where D is the diameter of G.*

The next result implies that the α-MPG can be randomly solved with any constant error bound in constant time, as long as the graph has large maximum degree.

Theorem 4.2 *Fix $\alpha < 1/2$. For any $\epsilon < 1$ and any graph G of maximum degree at least βn, for $\beta > 2\alpha$, there exists a randomized algorithm for the α-MPG, which errs with probability at most ϵ and has running time $O(\log(1/\epsilon))$ on every input.*

Proof: As H in Lemma 4.1 take the subgraph of G induced by any node of maximum degree together with its neighbors. This subgraph is connected and has diameter 2. □

5 Lower bounds for the α-majority problem on graphs

In this section we establish lower bounds on the performance of randomized algorithms for the α-MPG. The first lower bound shows that the complexity obtained in Theorem 4.2 for graphs of large maximum degree, cannot be improved, regardless of this degree. Of course, we cannot get the lower bound $\Omega(\log(1/\epsilon))$ for any error probability ϵ, when ϵ is a very fast decreasing function of the number n of nodes of the graph, since, for connected graphs, $n-1$ comparisons performed along edges of a spanning tree are always sufficient. Hence the best lower bound we can hope for in general is $\Omega(\min(n, \log(1/\epsilon)))$. We show that it actually holds.

Theorem 5.1 *Fix $\alpha < 1/2$. Let $G = (V, E)$ be an n-node graph. Every randomized algorithm for the α-MPG on G, that errs with probability at most ϵ, has expected running time $\Omega(min(n, \log(1/\epsilon)))$, for some input.*

Proof: We define a set $\mathcal{I}$ of input α-colorings for the α-MPG on G, and a (non-uniform) probability distribution on $\mathcal{I}$, such that any *deterministic* algorithm for the α-MPG on G, that errs with probability at most 2ϵ on this distribution, uses an expected number of $\Omega(\min(n, \log(1/\epsilon)))$ comparisons. By Yao's minimax principle, this implies our theorem.

Let $p = \alpha/2$. Consider the set $\mathcal{J}$ of all assignments of colors white or black to nodes in V. For a given assignment $J \in \mathcal{J}$, let $x(J)$ denote the number of black nodes in J. Define a probability distribution on $\mathcal{J}$ by the formula $\mathcal{P}'(J) = p^{x(J)}(1-p)^{n-x(J)}$. Distribution $\mathcal{P}'$ corresponds to random and independent coloring of every node: black with probability p and white with probability $1-p$. Let $\mathcal{I} = \{J \in \mathcal{J} | x(J) \leq \alpha n\}$ and let $q = \sum_{J \in \mathcal{I}} \mathcal{P}'(J)$. Define the probability distribution $\mathcal{P}$ on $\mathcal{I}$ by the formula $\mathcal{P}(J) = \mathcal{P}'(J)/q$. Set $\mathcal{I}$ consists of all α-colorings of G and probability distribution $\mathcal{P}$ is yielded by probability distribution $\mathcal{P}'$ by restricting it to $\mathcal{I}$ and normalizing. As usual, we extend $\mathcal{P}$ to subsets of $\mathcal{I}$ by taking the sum of distribution values over all elements of a subset.

By Lemma 2.1, there exists a positive constant b such that $1 - q \leq e^{-bn}$. Let d be a positive constant for which $e^{-bn} \leq p^{dn}/2$. There exists an ϵ_0 depending only on p, and a positive constant c, such that for $\epsilon < \epsilon_0$ the following inequality is satisfied

$$\frac{p^{2c\log(1/\epsilon)+1}}{4q} > \epsilon. \tag{1}$$

Fix such ϵ_0 and c. If $\epsilon \geq \epsilon_0$ then the theorem is true because $\Omega(\min(n, \log(1/\epsilon))) = \Omega(1)$ in this case. Hence we may assume $\epsilon < \epsilon_0$ and use inequality **??**.

Let $\gamma = \min((dn-1)/2, c\log(1/\epsilon))$. Fix a deterministic algorithm A for the α-MPG on G, that errs with probability at most 2ϵ on distribution $\mathcal{P}$. We will prove that $r(I) > \gamma$ for all $I \in \mathcal{I}$.

Suppose not. Hence for some input $I \in \mathcal{I}$ we have $r(I) \leq \gamma$. Fix such an input I. Let $U = V_I$ be the set of nodes involved in comparisons made by A on input I and let $u = u_I$ be the node presented by A as a solution on input I. Let $|U \cup \{u\}| = \delta$. By definition, $\delta \leq 2\gamma + 1$. Consider the following assignment $\mathcal{C}$ of colors white and black to nodes of the set $U \cup \{u\}$: if A is incorrect on I, then $\mathcal{C}$ is as in I; otherwise, $\mathcal{C}$ is reverse with respect to I (colors white and black are interchanged on $U \cup \{u\}$). Let $\mathcal{K}$ be the set of all inputs in $\mathcal{I}$ which have the assignment $\mathcal{C}$ of colors on the set $U \cup \{u\}$. We have $\mathcal{P}'(\mathcal{K}) \geq p^\delta - (1-q) \geq p^\delta - p^{dn}/2 \geq p^\delta/2$. Hence

$$\mathcal{P}(\mathcal{K}) \geq \frac{p^\delta}{2q} \geq \frac{p^{2c\log(1/\epsilon)+1}}{2q} > 2\epsilon.$$

For any $J \in \mathcal{K}$, we have $\mathcal{E}(A, J) = \mathcal{E}(A, I)$, hence A is incorrect on every input $J \in \mathcal{K}$. This implies that A errs with probability larger than 2ϵ, which is a contradiction.

Hence we must have $r(I) > \gamma$, for all $I \in \mathcal{I}$. This concludes the proof. □

Our next result shows that the upper bound given by Theorem 4.1 cannot be improved in general. For constant error probability ϵ we got running time linear in the diameter of the graph. We now show that for a large class of graphs, including d-dimensional grids and tori, this running time is optimal.

Fix a positive integer d. An m-node line is the graph with the set of nodes $V = \{v_1, ..., v_m\}$ and set of edges $E = \{\{v_i, v_{i+1}\} | i = 1, ..., m-1\}$. An m-node cycle has the same set of nodes and one edge more: $\{v_1, v_m\}$. A d-dimensional grid of size m is the graph $G_{d,m}$ which is the graph product of d copies of the m-node line. A d-dimensional torus of size m, denoted by $T_{d,m}$, is a d-dimensional grid of size m, supplemented with wrap around edges. More precisely, nodes of $T_{d,m}$ are all d-tuples $(x_1, \ldots, x_d)$, where $x_i = 0, ..., m-1$, for $i = 1, \ldots, d$. Each node $(x_1, x_2, \ldots, x_d)$ is connected to the $2d$ nodes $(x_1, \ldots, x_{i-1}, x_i \pm 1 \bmod m, x_{i+1} \ldots, x_d)$, for $i = 1, 2, \ldots, d$. Equivalently, $T_{d,m}$ is the graph product of d copies of an m-node cycle. Both the grid $G_{d,m}$ and the torus $T_{d,m}$ have $m^d = n$ nodes and diameter $\Theta(m)$.

Theorem 5.2 *Fix $\alpha < 1/2$ and $\epsilon < \alpha/8$. For any randomized algorithm R for the α-MPG on the grid $G_{d,m}$ or on the torus $T_{d,m}$, having error probability at most ϵ, there exists an input α-coloring I, such that the expected number of comparisons used by R on I is $\Omega(m)$.*

Proof: It is enough to prove this lower bound in the case of the torus $T_{d,m}$ because every algorithm for the grid $G_{d,m}$ works also for the torus $T_{d,m}$ (wrap around edges are simply not used).

We define a set $\mathcal{I}$ of input α-colorings for the α-MPG on $T_{d,m}$, such that any *deterministic* algorithm for the α-MPG on $T_{d,m}$, that errs with probability at most 2ϵ on the uniform

distribution over $\mathcal{I}$, uses an expected number of $\Omega(m)$ comparisons. By Yao's minimax principle, this implies our theorem.

Let $T_{d,m}$ be a d-dimensional torus of size m and let V be its set of nodes. Denote $m^d = n$. Given a node $v = (x_1, \ldots, x_d) \in V$, define $T_{d,m}(v)$ as the subgraph of $T_{d,m}$ induced by the nodes in the set $\{(y_1, \ldots, y_d) |\ x_i \leq y_i < x_i + \lfloor \sqrt[d]{\alpha n} \rfloor \bmod m,\ \text{for } i = 1, \ldots, d\}$. For every $v \in V$, define input I_v as the α-coloring such that the nodes in $T_{d,m}(v)$ are black and all the other nodes are white.

Define the set of inputs $\mathcal{I} = \{I_v |\ v \in V\}$. Notice that all inputs from $\mathcal{I}$ are α-colorings. Let $\mathcal{P}$ be the uniform distribution on $\mathcal{I}$. Let A be an arbitrary deterministic algorithm for the α-MPG that errs with probability at most 2ϵ. Fix $\gamma = \lfloor \beta m \rfloor$, where $\beta = \sqrt[d]{\alpha}/4$. Let $\mathcal{I}' = \{I \in \mathcal{I} \mid \text{ for every } e \in E_I(\gamma),\ \mathcal{L}_I(e) = 0\}$. In other words, $\mathcal{I}'$ is the set of inputs from $\mathcal{I}$ for which, in the first γ steps (or δ steps, if the running time of A on I is $\delta < \gamma$) algorithm A receives always answer 0. The following cases are possible: either there exists $I^* \in \mathcal{I}'$ such that $r(I^*) \leq \gamma$; or for all $I \in \mathcal{I}'$, we have $r(I) > \gamma$.

Suppose that there is $I^* \in \mathcal{I}'$ such that $r(I^*) \leq \gamma$. Let $\mathcal{E}(A, I^*) = (G_{I^*}, u_{I^*})$ be the execution of A on I^*. For every input $I \in \mathcal{I}'$, algorithm A receives the same answer 0 in all the first γ steps of its execution. Since there is an input $I^* \in \mathcal{I}'$ such that $r(I^*) \leq \gamma$, and since the algorithm is deterministic, then on all the inputs $I \in \mathcal{I}'$ the execution of A will be the same as on input I^*. That is, we must have $\mathcal{E}(A, I) = \mathcal{E}(A, I^*)$, for every $I \in \mathcal{I}'$. Denote by $u = u_{I^*}$ the solution provided by algorithm A on all the inputs $I \in \mathcal{I}'$, and let $F = E_{I^*}$ be the set of edges chosen by this algorithm on all the inputs $I \in \mathcal{I}'$.

Consider the set $\mathcal{B} = \{I \in \mathcal{I} |\ u \text{ is colored black on input } I \text{ and for every } e \in F, \mathcal{L}_I(e) = 0\}$. Clearly, $\mathcal{E}(A, I) = \mathcal{E}(A, I^*)$ on all the inputs $I \in \mathcal{B}$. Therefore algorithm A will err on all inputs $I \in \mathcal{B}$. In order to estimate the size of $\mathcal{B}$, we first observe that there are $\lfloor \sqrt[d]{\alpha n} \rfloor^d$ inputs in $\mathcal{I}$ that include u as a black node. For sufficiently large n, $\lfloor \sqrt[d]{\alpha n} \rfloor^d \geq \frac{3}{4} \cdot \alpha n$.

Denote by $\mathcal{B}_u$ this set of inputs. In order to obtain $\mathcal{B}$ from $\mathcal{B}_u$, we have to remove from $\mathcal{B}_u$ all inputs $I \in \mathcal{B}_u$ for which there is an edge $e \in F$ such that $\mathcal{L}_I(e) = 1$. For each edge $e \in F$, the number of inputs I such that $\mathcal{L}_I(e) = 1$ is at most

$$\mu = 2\lfloor \sqrt[d]{\alpha n} \rfloor^{d-1} \leq 2 \cdot (\sqrt[d]{\alpha n})^{d-1} = 2(\alpha n)^{\frac{d-1}{d}}.$$

(Indeed, fix an edge $e = \{v, w\}$ and consider an input I for which v and w are of different colors. For any such input, node v (or w) must be in the "hyperface" perpendicular to e of the d-dimensional box forming black nodes in I, and node w (resp. v) must be outside this box. There are two such hyperfaces, each of size $\lfloor \sqrt[d]{\alpha n} \rfloor^{d-1}$). Since $|F| = r(I^*) \leq \gamma$, the number of inputs that need to be removed is at most

$$\mu \cdot \gamma = 2(\alpha n)^{\frac{d-1}{d}} \lfloor \beta \sqrt[d]{n} \rfloor \leq 2\alpha^{\frac{d-1}{d}} \beta n = \alpha n/2.$$

Therefore, $|\mathcal{B}| \geq \frac{3}{4}\alpha n - \mu \cdot \gamma \geq \frac{3}{4}\alpha n - \alpha n/2 = \alpha n/4$. Hence, the probability that algorithm A errs on the uniform distribution $\mathcal{P}$, is at least

$$\frac{|\mathcal{B}|}{n} \geq \frac{\alpha n/4}{n} = \alpha/4 > 2\epsilon,$$

which is a contradiction.

Therefore, we have $r(I) > \gamma$, for all $I \in \mathcal{I}'$. Let us estimate the size of set $\mathcal{I}'$. For every $1 \leq i \leq \gamma$, define $Z_i = \{I \in \mathcal{I} |\, i \text{ is the first step at which } A \text{ gets answer } 1\}$. We have

$$\mathcal{I}' = \mathcal{I} \setminus \bigcup_{i=1}^{\gamma} Z_i.$$

Observe that for any $1 \leq i \leq \gamma$, we have $|Z_i| \leq \mu \leq 2(\alpha n)^{\frac{d-1}{d}}$. Therefore,

$$|\mathcal{I}'| \geq |\mathcal{I}| - \sum_{i=1}^{\gamma} |Z_i| \geq n - \mu \cdot \gamma \geq n - \alpha n/2.$$

Hence, since algorithm A spends more than $\gamma = \lfloor \beta m \rfloor$ steps for any input $I \in \mathcal{I}'$, the expected running time of algorithm A over $\mathcal{P}$ is

$$C(A, \mathcal{P}) \geq \frac{|I'| \cdot \gamma}{|I|} \geq \frac{(n - \alpha n/2)\lfloor \beta m \rfloor}{n} \in \Omega(m).$$

□

We conclude with the observation that the assumption of Theorem 4.2 cannot be weakened to only require maximum degree linear in the number of nodes: the large constant β, for maximum degree βn, turns out to be crucial to get a fast randomized algorithm.

Proposition 5.1 *Fix $\alpha < 1/2$. There exist n-node connected graphs G_n of maximum degree $\Theta(n)$, such that every randomized algorithm for the α-MPG on G_n, with sufficiently small constant error probability, has expected running time $\Omega(n)$, on some input.*

Proof: Fix $\alpha < 1/2$ and $\beta < \alpha$. Let $x = \lfloor \beta n \rfloor$ and $y = \lfloor \alpha n \rfloor - x$. Let G_n be the graph consisting of a cycle of size $n - x + 3$ with $x - 3$ additional nodes attached to one of its nodes, called v. Denote by Z the x-element set consisting of v and all of its neighbors. Let $\mathcal{J}$ be the set of all y-node segments of the cycle, and let $\mathcal{K} = \{Z \cup J | J \in \mathcal{J}\}$. Consider the set $\mathcal{I}$ of input α-colorings of G_n, for which all nodes in a set $K \in \mathcal{K}$ are black and all other nodes are white. Consider the uniform distribution on the set $\mathcal{I}$. An argument analogous to that from the proof of Theorem 5.2 (for $d = 1$, the d-dimensional torus is a cycle) shows that any deterministic algorithm with sufficiently small constant error probability uses expected time $\Omega(n)$ over the uniform distribution on $\mathcal{I}$. By Yao's minimax principle, this implies our result. □

6 Conclusion and open problems

We showed that randomization does not help significantly to solve the simple-majority problem on graphs: expected time linear in the number of nodes is still necessary if white nodes outnumber black nodes only by a constant. What happens in the more general case, if the difference is $o(n)$ for n-node graphs? Can the MPG be solved with constant error probability in sublinear expected time, e.g., if the algorithm knows that the difference is $\sqrt{n}$?

On the other hand, randomization drastically improves algorithm complexity for the α-majority problem on graphs, $\alpha < 1/2$, in some cases. For any constant error probability, it is possible to solve the α-MPG in time linear in the diameter of the graph, and this complexity is optimal for many graphs. In fact, for constant error probability, it is possible to solve the α-MPG in time linear in the diameter of a sufficiently large subgraph of the given graph. Thus, if a graph contains a sufficiently large subgraph of constant diameter (this is the case, e.g., for graphs of large maximum degree) then the α-MPG can be solved in constant time with constant error probability. However, for graphs of bounded maximum degree, constant diameter subgraphs are only of constant size, so our upper bound argument does not apply. Nevertheless, for some graphs of bounded maximum degree, the α-MPG can be solved in constant time with constant error probability. This is the case, e.g., for regular expander graphs, in view of the results from [5]. Indeed, take a random walk of constant length in a regular expander graph, and pick a node from the majority color among nodes visited in the walk. It follows from [5] that this node will be white with high probability. On the other hand, our lower bound for d-dimensional grids and tori shows that for some bounded degree graphs such a constant time algorithm does not exist. The following question remains open: for which graphs the α-MPG can be solved in constant time with constant error probability?

References

[1] M. Aigner, Variants of the majority problem, Discrete Applied Mathematics 137 (2004), 3-25.

[2] L. Alonso, E. M. Reingold, and R. Schott, Determining the majority, Information Processing Letters 47 (1993), 253-255.

[3] L. Alonso, E. M. Reingold, and R. Schott, The average-case complexity of determining the majority, SIAM Journal on Computing 26 (1997), 1-14.

[4] K. Y. Chwa and S. L. Hakimi, Schemes for fault-tolerant computing: A comparison of modularly redundant and t-diagnosable systems, Information and Control 49 (1981), 212-238.

[5] D. Gillman, A Chernoff bound for random walks on expander graphs, SIAM Journal on Computing 27 (1998), 1203-1220.

[6] T. Hagerup and C. Rub, A guided tour of Chernoff bounds, Information Processing Letters 33 (1989/90), 305-308.

[7] S. Kutten and D. Peleg, Fault-local distributed mending, Proc. 14th ACM Symposium on Principles of Distributed Computing (1995), 20-27.

[8] M. Malek, A comparison connection assignment for diagnosis of multiprocessor systems, Proc. 7th Symp. Comput. Architecture (1980), 31-35.

[9] F. P. Preparata, G. Metze, and R. T. Chien, On the connection assignment problem of diagnosable systems, IEEE Transactions on Electronic Computers 16 (1967), 848-854.

[10] M. E. Saks and M. Werman, On computing majority by comparisons, Combinatorica 11 (1991), 383-387.

[11] A. C-C. Yao, Probabilistic computations: Towards a unified measure of complexity, Proc. 18th Annual IEEE Conference on Foundations of Computer Science, FOCS (1977), 222-227.